# Contents of the Lecture

- 1. Introduction
- 2. Methods for I/O Operations
- 3. Computer Buses
- 4. Expansion Modules for Embedded Systems
- 5. Computer Displays
- 6. Graphics Adapters
- 7. Optical Discs

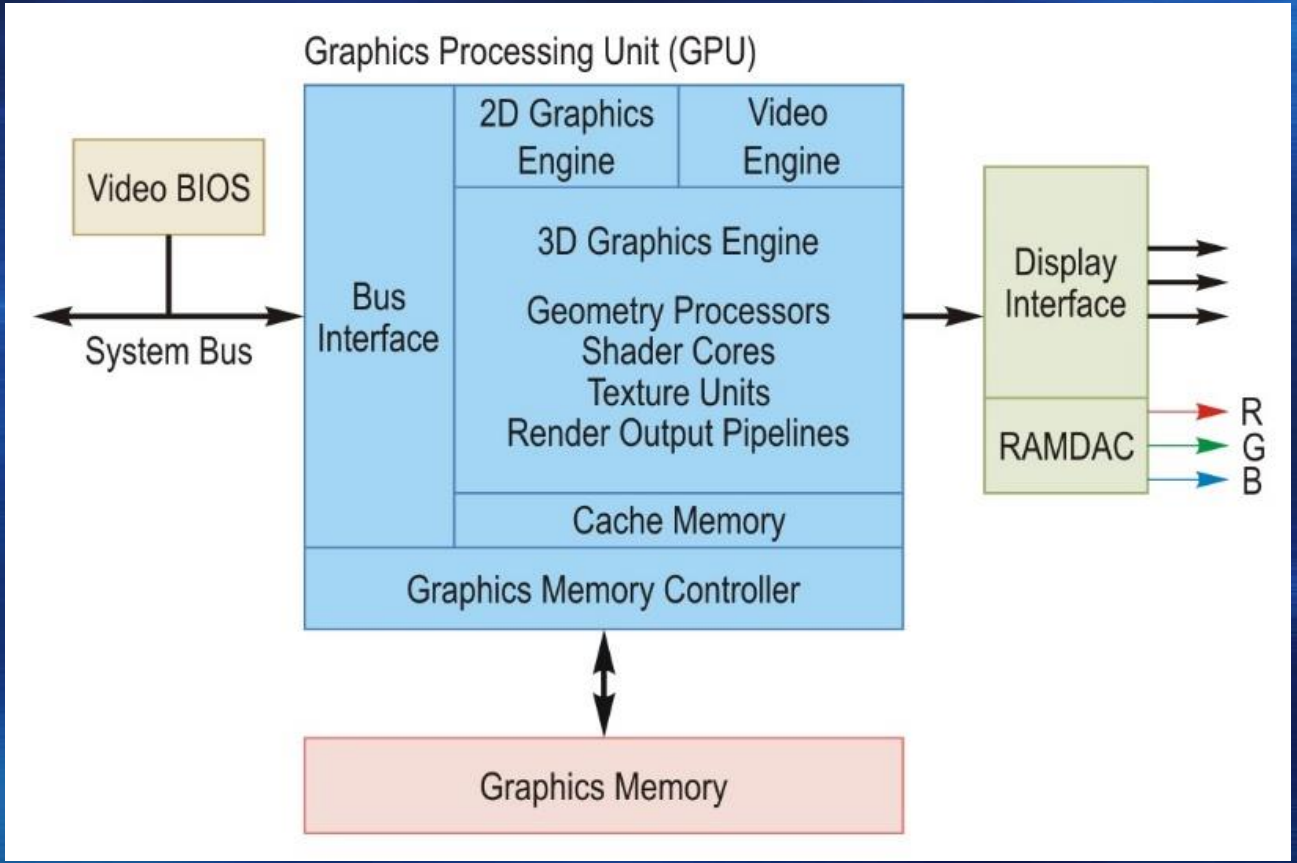
# 6. Graphics Adapters

- Structure of a Graphics Adapter
- Graphics Memory
- Graphics Processing Units
- Display Interfaces

# Structure of a Graphics Adapter (1)

- Graphics adapter: generates images for a display device
  - Additional functions:
    - Accelerated rendering of 2D and 3D graphics
    - Video decoding and encoding
    - Ability to connect multiple displays
  - Dedicated graphics adapter: resides on an expansion card
  - Integrated graphics adapter: integrated into a processor chip or system-on-chip (SoC)

# Structure of a Graphics Adapter (2)



# Structure of a Graphics Adapter (3)

- Video BIOS
  - Initializes the graphics adapter
  - Contains operating parameters for the graphics memory and GPU
  - Does not support all the features of the graphics adapter
- Graphics Processing Unit (GPU)
  - Implements the main functions of the graphics adapter

# Structure of a Graphics Adapter (4)

- Bus interface
  - Enables communication with the CPU and main memory
  - It should enable transfers in burst mode
- 2D Graphics engine
  - Implements functions to accelerate graphical user interfaces of OS and applications
  - Bit-block transfers (BitBlt): instructions for moving data in the framebuffer
  - Drawing geometric shapes
  - Filling polygons with a specified color or pattern

# Structure of a Graphics Adapter (5)

- Video engine
  - Provides hardware-accelerated video decoding, video encoding, and video transcoding
- Cache memory
  - Organized as a hierarchy of cache memories
  - Can be used as shared memory by the processing cores of the GPU
  - Fewer requests are needed to the graphics memory
- Graphics memory controller
  - Enables access to the graphics memory

# Structure of a Graphics Adapter (6)

- 3D Graphics engine
  - Performs the operations required for rendering 3D objects onto a 2D display
  - Two stages: geometry stage, rendering stage
  - Geometry stage: transformation, lighting, clipping; performed by geometry processors
  - Rendering stage: rasterization, shading, alpha blending; performed by shader cores
  - Texture units perform texture addressing, texture filtering, and texture mapping
  - Render output pipelines create the final pixels

# Structure of a Graphics Adapter (7)

- Graphics memory
  - Contains the framebuffer that holds the image displayed on the screen
  - Other data stored: textures, partial results for shader cores, compiled shader programs
- Display interface
  - Enables to transfer the graphics images to a display
  - May be a discrete component or integrated into the GPU chip

# Structure of a Graphics Adapter (8)

- May include a RAMDAC (*RAM Digital-to-Analog Converter*) circuit
  - Reads the digital image and converts it into analog signals
  - Only required for displays with analog inputs
- There are several standard display interfaces
- VGA (*Video Graphics Array*)
  - Analog interface
  - Designed for cathode ray tube (CRT) displays, but also used by some liquid crystal displays

# Structure of a Graphics Adapter (9)

- DVI (*Digital Visual Interface*)
  - Digital interface
  - DVI-I (digital and analog signals) or DVI-D (digital signals only) connector
- HDMI (*High-Definition Multimedia Interface*)
  - Digital interface that allows to send video data and audio data over the same cable
- DisplayPort
  - Digital interface for video and audio data
  - Uses a packet-based protocol

# 6. Graphics Adapters

- Structure of a Graphics Adapter
- Graphics Memory
- Graphics Processing Units
- Display Interfaces

# Graphics Memory

- Graphics Memory
  - Types of Graphics Memories
  - Graphics DDR6 Memory
  - Graphics DDR7 Memory
  - High Bandwidth Memory

# Types of Graphics Memories (1)

- Can be single-ported or dual-ported
- Single-ported graphics memory
  - The unique data port is used for refreshing the screen and for writing new data
- Dual-ported graphics memory
  - One of the ports is used for updating the images in memory
  - The second port has serial access and is used for refreshing the images on the screen

# Types of Graphics Memories (2)

- Examples of graphics memories
  - VRAM (*Video RAM*): dual-ported DRAM
  - WRAM (*Window RAM*): improved VRAM
  - SGRAM (*Synchronous Graphics RAM*): variant of SDRAM → can open two memory pages
  - DDR variants: DDR2, DDR3, DDR4, DDR5
  - GDDR (*Graphics Double Data Rate*)
    - Uses lower voltages and higher clock frequencies than DDR memories
    - Several generations: GDDR2 .. GDDR7

# Types of Graphics Memories (3)

- HBM (*High Bandwidth Memory*)
  - Interface for DRAM chips stacked vertically
  - Can achieve higher performance and lower power consumption than a 2D configuration
  - First generation: HBM (2013)
  - Second generation: HBM2 (2016)
  - Improved second generation: HBM2E (2018)
  - Third generation: HBM3 (2022), HBM3E (2023)
  - Fourth generation: HBM4 (2025)

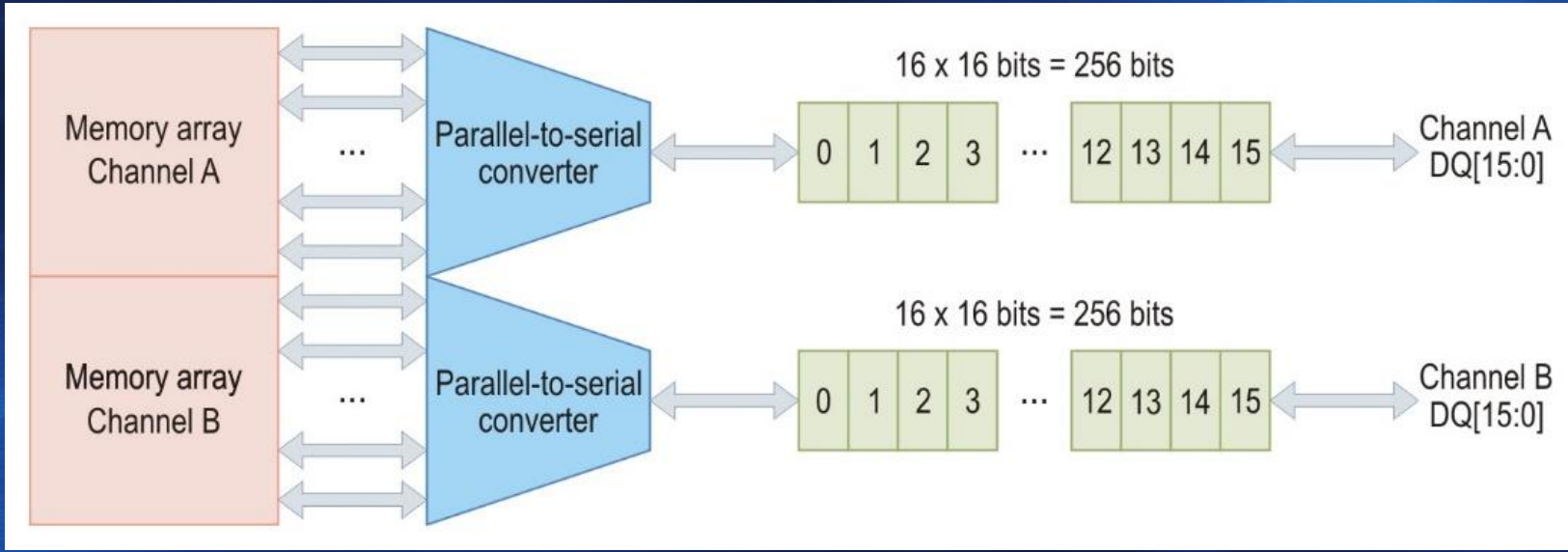
# Graphics Memory

- Graphics Memory
  - Types of Graphics Memories
  - Graphics DDR6 Memory
  - Graphics DDR7 Memory
  - High Bandwidth Memory

# Graphics DDR6 Memory (1)

- Combines high performance with stable operation and low implementation costs
- Memory organization: x32
- Uses 16*n* prefetching
  - Width of the internal data bus is 16x the width of the I/O interface
- The 32-bit I/O interface is divided into two independent 16-bit channels, A and B
  - On each channel, a read or write operation transfers 16 x 16 = 256 bits

# Graphics DDR6 Memory (2)




- Transfer rate for an access time of 1 ns to the memory array: 16 Gbits/s per pin
  - For a x32 memory device: 64 GB/s
- Max.: 24 Gbits/s per pin; 96 GB/s per device

# Graphics DDR6 Memory (3)

- Clock signals
  - Differential command clock signal (CK)
  - Two or four differential write clock signals (WCK0_A, WCK0_B, WCK1_A, WCK1_B)
    - Either two data bytes are aligned to a WCK signal, or each data byte is aligned to a separate WCK signal
    - QDR (*Quad Data Rate*) mode: the frequency of WCK signals ($f_{WCK}$) is doubled internally
    - DDR (*Double Data Rate*) mode: $f_{WCK}$ is not doubled

# Graphics DDR6 Memory (4)

- Techniques used
  - Data bus inversion
    - Reduces the number of zero bits transmitted
    - Indicated with a DBI# signal for each byte
    - Transmission lines have high-level termination $\rightarrow$ power dissipation is reduced
  - Command/address (CA) bus inversion
  - Signal training
    - Phase adjustment of clock, data, and address signals

# Graphics DDR6 Memory (5)

- Address training: alignment of address lines to the CK signal
- Data training: alignment of the data lines to the corresponding WCK signal
- Alignment of each WCK signal to the CK signal
- A “hidden” data re-training is possible
- Calibration: improves the reliability
- Auto-calibration: drive strength, termination impedance
- Software-controlled adjustment

# Graphics DDR6 Memory (6)

- Error detection
  - Dedicated EDC (*Error Detection Code*) pins for sending CRC codes to the memory controller
  - For each channel: a 16-bit CRC code for the data signals and DBI# signals of a burst transfer block
    - CRC code: concatenation of two 8-bit CRC codes, for the first half and second half of the block
  - Allows full detection of random single-bit, double-bit, and triple-bit errors, and >99% detection of other random errors

# Graphics DDR6 Memory (7)

- Power management features
  - Scalable clock frequency: from 50 MHz (400 Mbits/s) to the maximum frequency
  - Low-power mode for the DRAM core
  - Increasing the termination impedance at slower data rates
  - Self-refresh and hibernate self-refresh modes
  - Low supply voltage: 1.35 V
  - Data and address bus inversion

# Graphics Memory

- Graphics Memory
  - Types of Graphics Memories
  - Graphics DDR6 Memory
  - Graphics DDR7 Memory
  - High Bandwidth Memory

# Graphics DDR7 Memory

- Uses 32*n* prefetch architecture
  - Four independent 8-bit channels
- Pulse Amplitude Modulation (PAM)
  - Three signal voltage levels (PAM3): 3 bits transmitted over 2 cycles
  - More efficient than the NRZ (*Non-Return to Zero*) modulation used by GDDR6
- Transfer rate: 32 Gbits/s per pin (Gen1 devices); up to 48 Gbits/s per pin
  - 128 GB/s per device; up to 192 GB/s per dev.

# Graphics Memory

- Graphics Memory
  - Types of Graphics Memories
  - Graphics DDR6 Memory
  - Graphics DDR7 Memory
  - High Bandwidth Memory

# High Bandwidth Memory (1)

- Consists of one or more memory stacks
  - Up to 8, 12, or 16 DRAM dies
  - Logic die containing a memory controller
  - Vertical interconnections
- Connection to a CPU/GPU/SoC via physical interfaces and a silicon die → interposer
- The memory stacks, CPU/GPU/SoC die, and silicon die are encapsulated in a single package

# High Bandwidth Memory (2)

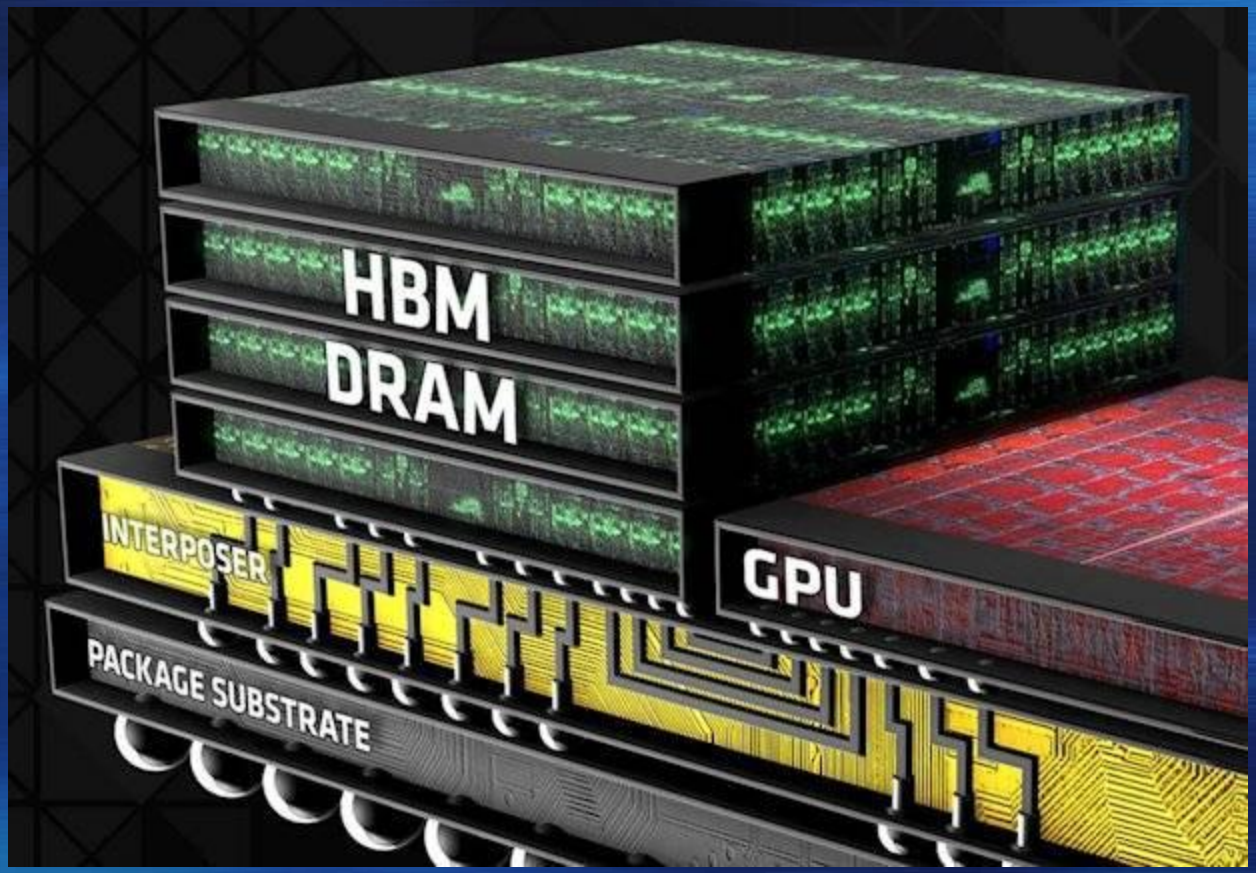



Image credit Advanced Micro Devices Inc.

# High Bandwidth Memory (3)

- First generation (HBM)
  - Operation at 1.3 V, 500 MHz
  - Transfer rate: 1 GT/s per line (pin)
  - Each DRAM die in the stack communicates via 128-bit channels
  - Eight channels per stack
    - Interface width: 8 x 128 = 1024 bits
    - Bandwidth: 128 GB/s per memory device
  - Memory capacity: up to 4 GB per device

# High Bandwidth Memory (4)

- Second generation (HBM2)
  - Operation at 1.2 V, 1 GHz
  - Transfer rate: 2 GT/s per pin
  - Bandwidth: 256 GB/s per memory device
  - Memory capacity: up to 8 GB per device
- Improved second generation (HBM2E)
  - Transfer rate: 3.2 GT/s per pin
  - Bandwidth: 409.6 GB/s per memory device
  - Up to 12 dies, up to 24 GB per device

# High Bandwidth Memory (5)

- Third generation (HBM3)
  - 16 channels of 64 bits (16 x 64 = 1024)
  - Transfer rate: 6.4 GT/s per pin
  - Bandwidth: 819.2 GB/s per memory device
  - Up to 48 GB (12 dies), 64 GB (16 dies)
- Improved third generation (HBM3E)
  - Transfer rate: 8 GT/s per pin (1 TB/s per dev.)
  - Micron: 9.2 GT/s (1.15 TB/s per device)
  - Samsung: 9.8 GT/s (1.22 TB/s per device)

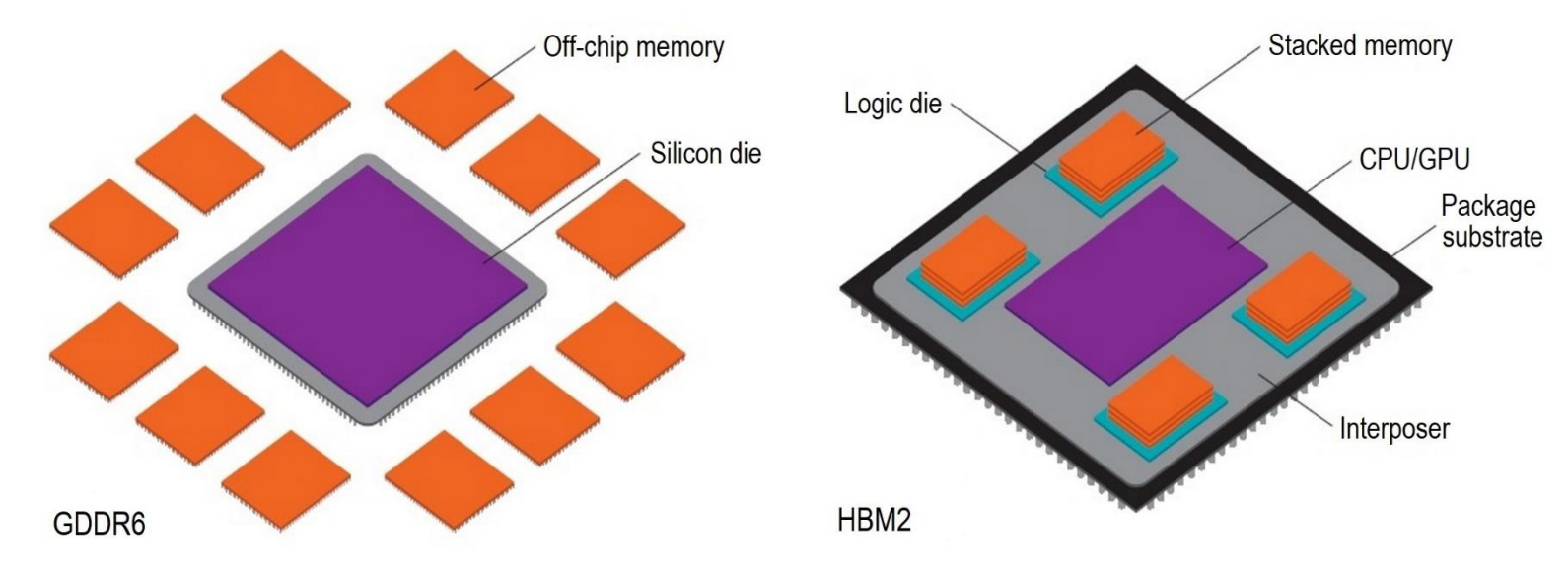
# High Bandwidth Memory (6)

- Fourth generation (HBM4)
  - Operating voltage: 1.0 V or 1.05 V
  - The number of channels per memory stack has been doubled, to 32
    - Interface width: 32 x 64 bits = 2048 bits
  - Transfer rate: 8 GT/s per pin
  - Bandwidth: 2 TB/s per memory device
  - Compatibility with existing HBM3 controllers
  - 4, 8, 12, or 16 dies, up to 64 GB per device

# High Bandwidth Memory (7)

- Advantages compared to a GDDR memory
  - Higher bandwidth
    - HBM3: 8.5x higher compared to GDDR6
    - HBM4: 10.7x higher compared to GDDR7
  - Reduces power consumption
  - Requires less space
    - A GDDR6 memory with 4 chips requires an area 20x larger than a stack of 4 HBM dies
- Disadvantage: higher cost

# High Bandwidth Memory (8)




Original image credit Advanced Micro Devices Inc.

# 6. Graphics Adapters

- Structure of a Graphics Adapter
- Graphics Memory
- Graphics Processing Units
- Display Interfaces

# Graphics Processing Units

- Graphics Processing Units
  - GPU Overview
  - GPU Pipeline
  - Data-Parallel Architecture
  - GPGPU Computing
  - CUDA

# GPU Overview (1)

- GPU – *Graphics Processing Unit*
- Dedicated graphics processors for PCs, game consoles, smartphones
  - Initially used to accelerate the rendering stage for 3D graphics (e.g., texture mapping)
  - Later also used to accelerate the geometry computations (rotation, translation)
- GPUs contain many shader cores, texture units, render output pipelines

# GPU Overview (2)

- Shader cores (shaders)
  - Operate on vertices, primitives, or pixels
  - Execute shader programs
- GPUs of previous generations contained specialized shader cores
- Vertex shader cores
  - Used to add special effects to 3D objects
  - Operate on the objects' vertex data: color, texture, lighting
  - Examples: animation, surface deformation

# GPU Overview (3)

- Geometry shader cores
  - Not always present
  - Receive as inputs the vertices for a primitive assembled in a previous pipeline stage
  - May add, remove, or change primitives
- Pixel/fragment shader cores
  - Used to add light, shade, and texture effects to individual pixels
  - Enable to generate complex, realistic scenes

# GPU Overview (4)

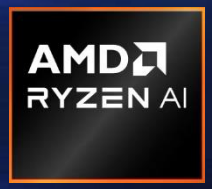
- Unified shader cores
  - Able to perform various shading operations (vertex, geometry, pixel/fragment)
  - GPUs contain an array of unified shader cores and a dynamic scheduling unit
  - The unified shader cores enable a more flexible use of the hardware resources
- Compute shader cores
  - Provide general-purpose compute resources
  - Take advantage of the large number of shader pipelines available

# GPU Overview (5)

- GPUs can be dedicated or integrated
- Dedicated GPUs
  - Used in graphics cards → contain a graphics memory dedicated for the GPU
  - Communicate with the CPU via the system bus, usually a PCI Express bus
  - Examples
    - AMD Radeon RX 9000 (e.g., RX 9070)
    - NVIDIA GeForce RTX 50 (e.g., GeForce RTX 5080)


# GPU Overview (6)

- Integrated GPUs
  - Are integrated into a chipset or SoC
  - Use a portion of the system memory
  - Have lower performance compared to dedicated GPUs
  - Examples
    - Intel $X^e$ (e.g., Iris $X^e$ Graphics)
    - AMD Radeon 890M in AMD Ryzen AI 9 HX 370 series SoCs
    - ARM Mali (e.g., Mali-G715 in the Google Tensor G4 chipsets)


# GPU Overview (7)

- Hybrid solutions
  - Hybrid graphics cards
    - Share memory with the system, but also have a smaller dedicated memory
    - Technologies have been developed for accessing the system memory (e.g., via a PCI Express bus)
  - Integrating an embedded DRAM (eDRAM) in the same package as the GPU
    - Examples: Intel Iris Graphics series of GPUs, with 64 MB or 128 MB of eDRAM

# GPU Overview (8)

- Unified memory architecture (UMA)
  - Used by SoCs and game consoles
  - The GPU can use any part of the system memory for various buffers and textures
  - Examples: Intel SoCs
    - The integrated GPU has its own set of L1 and L2 cache memories, and an L3 cache memory
    - Last-level cache (LLC): shared between the GPU and the CPU cores
  - Hierarchy of cache memories: reduces memory latency and bandwidth usage

# GPU Overview (9)

- The design of GPUs was influenced by 2D and 3D graphics library specifications
  - Manufacturers created implementations using hardware acceleration
  - Special tests qualified an implementation as compliant with a certain specification
  - A correspondence emerged between features offered by hardware acceleration and those offered by the graphics libraries
  - Examples: OpenGL, DirectX, Vulkan

# GPU Overview (10)


- OpenGL (*Open Graphics Library*)
  - Multi-platform and multi-language specification
  - Maintained by the Khronos Group
  - Provides functions for specifying shader programs and their data
  - OpenGL ES: subset for rendering 2D and 3D graphics on embedded systems
  - WebGL (*Web Graphics Library*): rendering within a web browser using JavaScript

# GPU Overview (11)


- DirectX
  - Provided by Microsoft Corp. for Windows platforms and Xbox game consoles
  - It includes the Direct2D, Direct3D, and DirectCompute APIs
  - DirectX 12 (2015) is a radical redesign of the API for modern GPU architectures
    - Significantly reduces the driver overhead
    - Improves the parallelism by using more CPU cores for graphics

# GPU Overview (12)

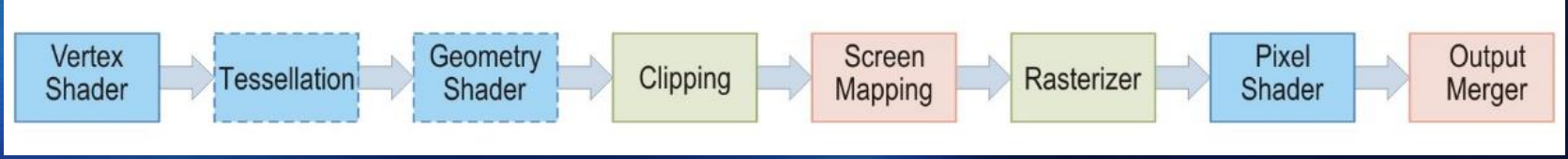

- Vulkan
  - Multi-platform API for 3D graphics and general-purpose computing
  - Original version donated by AMD to the Khronos Group
  - Can take full advantage of modern GPUs and multi-core CPUs
  - Offers better support for multithreaded CPUs and reduces the load on the CPU
  - Supports almost all the major OSs

# Graphics Processing Units

- Graphics Processing Units
  - GPU Overview
  - GPU Pipeline
  - Data-Parallel Architecture
  - GPGPU Computing
  - CUDA

# GPU Pipeline (1)

- Logical pipeline
- Divided into several hardware stages
  - Programmable stages (light blue)
  - Configurable stages (pink)
  - Fixed-function stages (light green)
  - Optional stages: tessellation, geometry shader



# GPU Pipeline (2)

- Vertex shader
  - Operates on individual vertices of primitives (points, lines, triangles)
  - Used to modify or create values associated with each vertex: position, color, texture
  - Implements geometry processing operations
    - Model transform: translation, rotation, scaling
    - View transform: positioning the view camera at the origin of the coordinate system
    - Lighting: determining the effect of light sources on an object's material

# GPU Pipeline (3)

- Tessellation
  - Enables to render curved surfaces
  - Converts the description of a high-order surface into a set of triangles
  - Advantages:
    - The description of a curved surface is more compact
    - Saves memory and reduces the bandwidth
    - Enables scalable rendering techniques → level of detail dependent on the position of the camera

# GPU Pipeline (4)

- Geometry shader
  - Enables to convert primitives into other primitives
  - Inputs: vertices for a single primitive
  - Outputs: zero or multiple vertices forming a single topology (e.g., triangle strip)
- Clipping
  - Discards primitives that are outside of the view volume and eliminates the parts of primitives that fall outside the view volume

# GPU Pipeline (5)

- Screen mapping
  - Transforms the 3D coordinates of primitives into 2D coordinates on the screen
- Rasterizer
  - Determines all the pixels that are inside a primitive being rendered
  - Triangle setup: computes the data required by a particular rasterization algorithm
  - Triangle traversal: determines the pixels that are inside a triangle $\rightarrow$ generates fragments

# GPU Pipeline (6)

- Pixel shader
  - Performs shading computations on each pixel
  - Inputs: results from the vertex shader, interpolated by the triangle traversal stage
  - Outputs: color, opacity value, and depth value for a pixel
  - Texturing: changing the appearance of a surface using a texture image or function
  - Texture filtering: reduces unwanted effects when a texture is smaller or larger than the pixel area being textured

# GPU Pipeline (7)

- Output merger
  - Combines the pixel color with the color currently stored in the color buffer
  - Also called raster operations pipeline (ROP)
  - Performs visibility tests using the z-buffer
  - For each pixel, a transparency (alpha) value is also stored
  - The color buffer stores an RGBA (RGBα) color and alpha value for each pixel
  - Alpha blending: mixing a transparent object's color with the color of the object behind it

# Graphics Processing Units

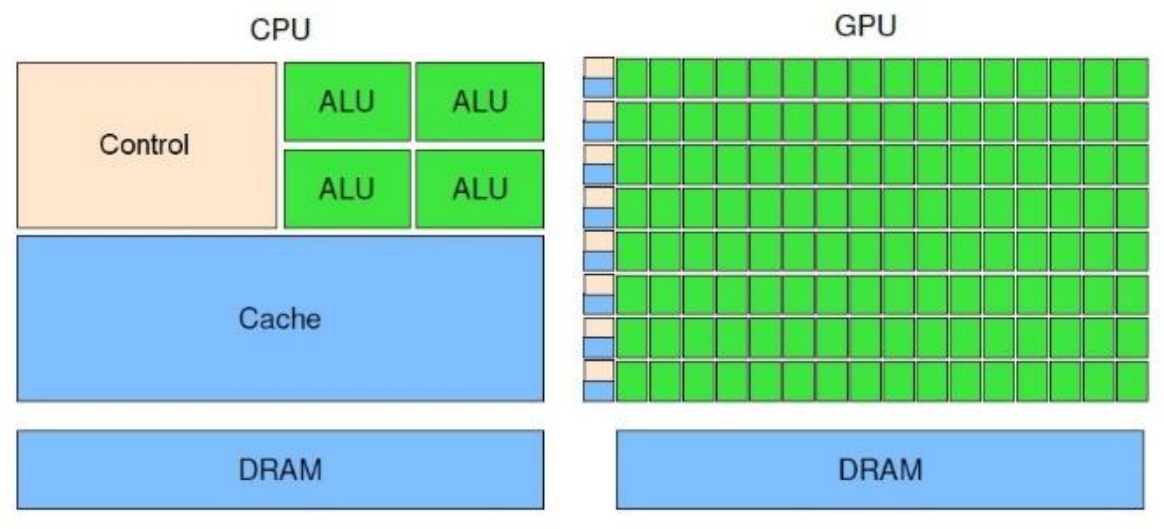
- Graphics Processing Units
  - GPU Overview
  - GPU Pipeline
  - Data-Parallel Architecture
  - GPGPU Computing
  - CUDA

# Data-Parallel Architecture (1)

- The GPU architecture is different from the CPU architecture
  - GPU architecture: optimized for throughput
    - Throughput: number of operations performed in a given time
    - A GPU has thousands of simple processing cores
  - CPU architecture: optimized for latency
    - Latency: delay from initiating a command until its effect becomes detectable
    - A CPU only has a few processing cores
    - A CPU dedicates a large area for cache memories

# Data-Parallel Architecture (2)

- CPUs implement complex techniques: instruction pre-fetching, branch prediction
- The control unit of a CPU becomes complex
- GPU: high throughput, but high latency
- CPU: low latency, but low throughput



# Data-Parallel Architecture (3)

- A GPU uses parallel execution at much more extended scale than a CPU
  - Shader cores execute many threads in parallel
  - Threads are largely independent of each other
  - Type of parallelism: data parallelism
  - Architecture: SIMD (*Single Instruction, Multiple Data*), SIMT (*Single Instruction, Multiple Thread*)
- A GPU thread is simple, and its creation has little overhead
  - Uses a small local memory and registers

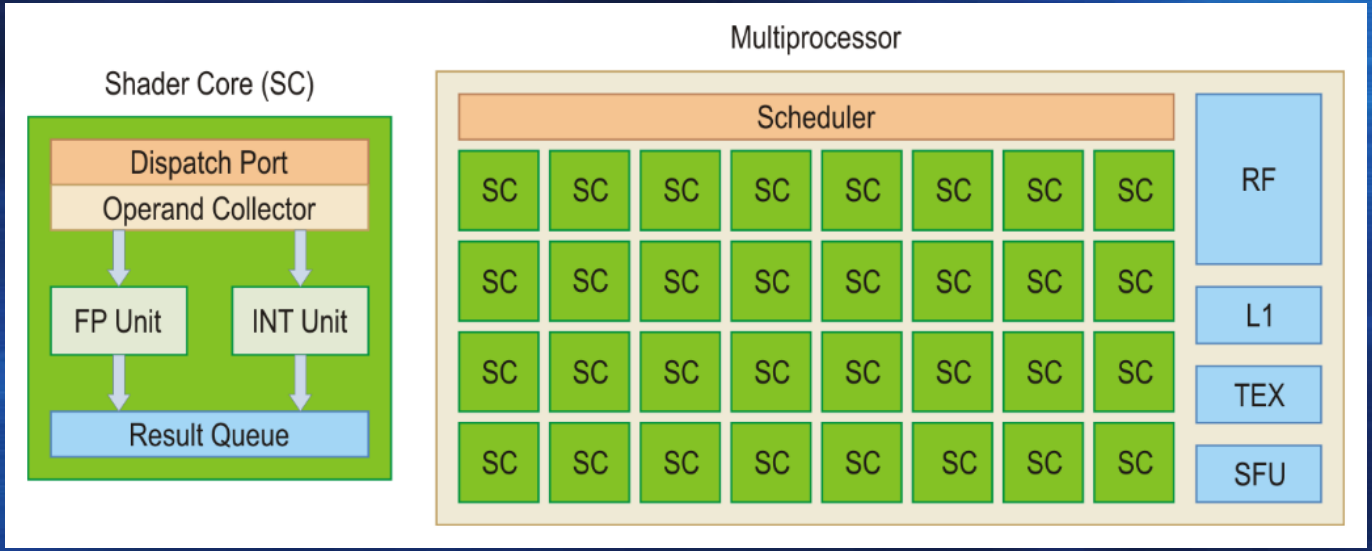
# Data-Parallel Architecture (4)

- Threads that execute the same shader program are grouped into batches of threads (e.g., *warps*)
  - A memory fetch instruction is encountered by all threads in a batch at the same time
  - All the threads in the batch are suspended and the batch is swapped out for another batch
  - Swapping between two batches of threads is fast $\rightarrow$ no data within each thread is affected
  - The fast swapping between batches of threads hides memory latencies

# Data-Parallel Architecture (5)

- Unified shader core
  - Executes a shader program
  - Components: instruction decoder, operand collector, ALUs, result collector
  - Names: stream processor (AMD); pipeline (Imagination Techn.); CUDA core (NVIDIA)
  - Several shader cores (e.g., 32) are grouped into a SIMD engine → multiprocessor (MP)
    - Names: compute unit (AMD); execution unit (Intel), streaming multiprocessor (NVIDIA)

# Data-Parallel Architecture (6)




- Scheduler: assigns the threads to the shader cores
- RF: register file
- L1: level-1 cache memory
- TEX: texture units
- SFU: special-function unit

# Summary (1)

- A graphics adapter generates images for a display device
  - Important components: graphics processing unit (GPU), graphics memory, display interface
- The graphics memory contains the framebuffer, other data buffers, and shader programs
  - Can be single-ported or dual-ported
  - Dual-ported memory: updating the images and refreshing the screen can be performed in parallel
  - Examples of graphics memory technologies: GDDR, HBM

# Summary (2)

- The GDDR6 memory has advanced features for high performance and stable operation
  - Data and address bus inversion; signal training; calibration; error detection; power management
- The HBM3 and HBM4 memories have high bandwidths and require a reduced area
- The GPU accelerates the geometry and rendering stages of the 3D graphics pipeline
  - Contains many shader cores, texture units, and raster operation pipelines
  - Current GPUs contain unified shader cores

# Summary (3)

- GPUs can be dedicated or integrated into a chipset
- The design of GPUs was influenced by 2D and 3D graphics library specifications
- The GPU pipeline contains programmable, configurable, and fixed-function stages
- The GPU architecture is different from the CPU architecture
- GPUs use SIMD/SIMT architectures and data parallelism
- Threads are grouped into batches and are swapped out rapidly to hide memory latencies

# Concepts, Knowledge (1)

- Structure of a graphics adapter
- Components of the graphics processing unit
- Types of graphics memories
- Features of the GDDR6 memory
- Techniques used by the GDDR6 memory
- Power management features of the GDDR6 memory
- Generations of HBM memory
- Advantages of HBM memory

# Concepts, Knowledge (2)

- Types of shader cores of a GPU
- Dedicated and integrated GPUs
- Stages of the GPU pipeline
- Differences between the GPU architecture and the CPU architecture
- Data parallelism provided by the GPU architecture
- Structure of a unified shader core
- Structure of a GPU multiprocessor